

PURE PRODUCT POLYNOMIALS AND THE PROUHET-TARRY-ESCOTT PROBLEM

ROY MALTBY

ABSTRACT. An n -factor pure product is a polynomial which can be expressed in the form $\prod_{i=1}^n (1 - x^{\alpha_i})$ for some natural numbers $\alpha_1, \dots, \alpha_n$. We define the norm of a polynomial to be the sum of the absolute values of the coefficients. It is known that every n -factor pure product has norm at least $2n$. We describe three algorithms for determining the least norm an n -factor pure product can have. We report results of our computations using one of these algorithms which include the result that every n -factor pure product has norm strictly greater than $2n$ if n is 7, 9, 10, or 11.

1. INTRODUCTION

For any $n \in \mathbf{N}$, an n -factor pure product is a polynomial which can be expressed in the form

$$\prod_{i=1}^n (1 - x^{\alpha_i})$$

for some $\alpha_1, \dots, \alpha_n \in \mathbf{N}$. The 1-norm of a polynomial is the sum of the absolute values of its coefficients. That is, for any polynomial $p(x) = \sum_{i=0}^d a_i x^i$, the 1-norm is $\sum_{i=0}^d |a_i|$ which we denote by $\|p(x)\|_1$. Actually, since the 1-norm is the only norm we use in this paper, we will usually just say “norm” rather than “1-norm”. For each $n \in \mathbf{N}$, define

$$A_1(n) = \min_{\alpha_1, \dots, \alpha_n \in \mathbf{N}} \left\| \prod_{i=1}^n (1 - x^{\alpha_i}) \right\|_1.$$

Old results tell us that $A_1(n) \geq 2n$ for every $n \in \mathbf{N}$. A simple proof of this is repeated in [BI94]. Before our research, it was known that $A_1(n) = 2n$ for $n \in \{1, 2, 3, 4, 5, 6, 8\}$ and no other values of $A_1(n)$ were known. In [BI94], Borwein and Ingalls conjectured that $A_1(7) = 16$, which we verify in this paper, and they posed the following three (not entirely distinct) problems, which we solve in this paper.

5. Show that there is no 7-factor pure product of norm 14.

Received by the editor October 16, 1995 and, in revised form, June 19, 1996.

1991 *Mathematics Subject Classification.* Primary 11Y50, 11B75.

Key words and phrases. Prouhet-Tarry-Escott Problem, Tarry-Escott Problem, Erdős-Szekeres Problem.

©1997 by the author

6b. Prove

$$\min_{\alpha_1, \dots, \alpha_n} \left\| \prod_{i=1}^n (1 - x^{\alpha_i}) \right\|_1 > 2n$$

for some n . (Problem 5 is the $n = 7$ case of this.)

8. Find a true algorithm, even an impractical one, that determines if there is a 7-factor pure product of norm 14.

We will describe three algorithms for computing $A_1(n)$, the most efficient of which we have implemented to find that $A_1(7) = 16$, $A_1(9) = 20$, $A_1(10) = 24$, and $A_1(11) \geq 24$.

A problem on which pure products have some bearing is the Prouhet-Tarry-Escott Problem. We use square brackets to delimit a *list*, and we call two lists (or, more precisely, *k-lists*) $[a_1, \dots, a_k]$ and $[b_1, \dots, b_k]$ equal if (a_1, \dots, a_k) is a permutation of (b_1, \dots, b_k) . That is, a list is like a set except that repeated elements are allowed, and a list is like a tuple except that the order of entries does not matter. (Some authors call a list a *multiset*.) Suppose we have two unequal lists of integers $[a_1, \dots, a_k]$ and $[b_1, \dots, b_k]$ such that

$$\sum_{i=1}^k a_i^r = \sum_{i=1}^k b_i^r$$

for $r = 1, 2, \dots, d$. Then we say that $[a_1, \dots, a_k]$ and $[b_1, \dots, b_k]$ form a *multigrade* of size k and degree d . The Prouhet-Tarry-Escott Problem is to find multigrades of the smallest possible size for each degree. In Section 5 we repeat old results showing that the size of any multigrade is strictly greater than its degree. For each degree up to $d = 9$, multigrades of size $d + 1$ are known, and it is conjectured (cf. [BI94, p. 10]) that such multigrades exist for all degrees. The connection to pure products is that any n -factor pure product of norm $2k$ can be used to construct a multigrade of degree $n - 1$ and size k . This conjecture is the reason for being particularly interested in determining the values of n for which $A_1(n) = 2n$.

The problem of determining the least possible 1-norm of an n -factor pure product is also related to a problem of Erdős and Szekeres. Their problem [ES58] was exactly the same except that instead of the 1-norm of a polynomial, they used the ∞ -norm which is defined by

$$\left\| \sum_{i=0}^d a_i x^i \right\|_{\infty} := \sup_{\{z \in \mathbf{C}: |z|=1\}} \left| \sum_{i=0}^d a_i z^i \right|,$$

where \mathbf{C} is the set of complex numbers and $|z|$ denotes the modulus (*i.e.*, absolute value) of the complex number z . For any polynomial $\sum_{i=0}^d a_i x^i$, the 1-norm and the ∞ -norm are related by these inequalities:

$$\frac{\left\| \sum_{i=0}^d a_i x^i \right\|_1}{\sqrt{d+1}} \leq \left\| \sum_{i=0}^d a_i x^i \right\|_{\infty} \leq \left\| \sum_{i=0}^d a_i x^i \right\|_1.$$

2. PROPERTIES OF THE 1-NORM

The problem of finding pure products of small 1-norm has an equivalent statement in terms of certain equalities among sums of subsets of a set of natural numbers. The following easy lemmas describe this connection. We use \mathbf{Z} to denote the

set of integers, \mathbf{N} to denote the set of natural numbers, and for each $n \in \mathbf{N}$, we write \mathbf{n} to denote the set $\{1, \dots, n\}$.

Lemma 2.1. *Let n and $\alpha_1, \dots, \alpha_n$ be natural numbers. Let $d = \sum_{i=1}^n \alpha_i$ and $a_0, \dots, a_d \in \mathbf{Z}$ so that, as polynomials, $\prod_{i=1}^n (1 - x^{\alpha_i}) = \sum_{k=0}^d a_k x^k$. For $k = 0, \dots, d$, let*

$$\mathcal{E}_k = \{I \subseteq \mathbf{n} : |I| \text{ even, } \sum_{i \in I} \alpha_i = k\},$$

$$\mathcal{O}_k = \{I \subseteq \mathbf{n} : |I| \text{ odd, } \sum_{i \in I} \alpha_i = k\}.$$

Then $a_k = |\mathcal{E}_k| - |\mathcal{O}_k|$ for $k = 0, \dots, d$. □

The proof is obvious. We will use Lemma 2.1 frequently without explicitly mentioning it.

The following lemma provides a useful condition for ending a branch in the search tree of our algorithm in Section 5, at least for small values of n .

Lemma 2.2. *In Lemma 2.1, if n is odd, then $a_k = -a_{d-k}$ for $k = 0, \dots, d$. If n is even, then $a_k = a_{d-k}$ for $k = 0, \dots, d$.*

Proof. Suppose n is odd. Observe that for $k = 0, \dots, d$,

$$\mathcal{E}_{d-k} = \{\mathbf{n} \setminus I : I \subseteq \mathcal{O}_k\}$$

and

$$\mathcal{O}_{d-k} = \{\mathbf{n} \setminus I : I \subseteq \mathcal{E}_k\}.$$

So the result is clear from Lemma 2.1. In the special case $k = \frac{d}{2}$, this tells us that $a_{\frac{d}{2}} = 0$.

Now suppose n is even. Then, for $k = 0, \dots, d$,

$$\mathcal{E}_{d-k} = \{\mathbf{n} \setminus I : I \subseteq \mathcal{E}_k\}$$

and

$$\mathcal{O}_{d-k} = \{\mathbf{n} \setminus I : I \subseteq \mathcal{O}_k\}.$$

Again, the result is clear from Lemma 2.1. □

The following lemma might be considered a more comprehensive statement of Lemma 2.1. One of the simplest implications of this lemma is that the norm of every pure product is even.

Lemma 2.3. *Let $\alpha_1, \dots, \alpha_n \in \mathbf{N}$. Let \mathcal{P} be a set so that for every $P \in \mathcal{P}$, $P = \{I, J\}$ for some $I, J \subseteq \mathbf{n}$ such that $|I|$ is even, $|J|$ is odd, and $\sum_{i \in I} \alpha_i = \sum_{j \in J} \alpha_j$. Furthermore, choose \mathcal{P} so that $P \cap Q = \emptyset$ for all distinct $P, Q \in \mathcal{P}$. Suppose \mathcal{P} is maximal with these properties. Then $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1 = 2^n - 2|\mathcal{P}|$.*

Proof. Put $d = \sum_{i=1}^n \alpha_i$ and for $k = 0, \dots, d$, put $\mathcal{P}_k = \{\{I, J\} \in \mathcal{P} : \sum_{i \in I} \alpha_i = \sum_{j \in J} \alpha_j = k\}$. Then each $\{I, J\} \in \mathcal{P}_k$ such that $|I|$ is even and $|J|$ is odd has $I \in \mathcal{E}_k$ and $J \in \mathcal{O}_k$, using the notation of Lemma 2.1. Furthermore, for each $k = 0, \dots, d$, $|a_k| = |\mathcal{E}_k| + |\mathcal{O}_k| - 2 \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} = |\mathcal{E}_k| + |\mathcal{O}_k| - 2|\mathcal{P}_k|$. So $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1 = \sum_{k=0}^d |a_k| = \sum_{k=0}^d (|\mathcal{E}_k| + |\mathcal{O}_k| - 2|\mathcal{P}_k|) = 2^n - 2|\mathcal{P}|$. □

Our most extensive use of these lemmas is in the algorithm in Section 5, but they also give us the following result.

Theorem 2.4. *Let $n > 1$ and let $\alpha_1, \dots, \alpha_n \in \mathbf{N}$. Let $d = \sum_{i=1}^n \alpha_i$. If n is even or d is odd, then $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$ is a multiple of 4.*

Proof. Assign $d, a_0, \dots, a_d, \mathcal{E}_0, \dots, \mathcal{E}_d, \mathcal{O}_0, \dots, \mathcal{O}_d$ as in Lemma 2.1. Then

$$\begin{aligned} \left\| \prod_{i=1}^n (1 - x^{\alpha_i}) \right\|_1 &= \sum_{k=0}^d |a_k| = \sum_{k=0}^d \left| |\mathcal{E}_k| - |\mathcal{O}_k| \right| \\ &= \sum_{k=0}^d (|\mathcal{E}_k| + |\mathcal{O}_k| - 2 \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\}) \\ &= 2^n - 2 \sum_{k=0}^d \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\}. \end{aligned}$$

From the proof of Lemma 2.2, it is apparent that for $k = 0, \dots, d$, $\{|\mathcal{E}_k|, |\mathcal{O}_k|\} = \{|\mathcal{E}_{d-k}|, |\mathcal{O}_{d-k}|\}$. So if d is odd, then

$$\begin{aligned} \sum_{k=0}^d \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} &= \sum_{k=0}^{\lfloor \frac{d}{2} \rfloor} \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} + \sum_{k=\lceil \frac{d}{2} \rceil}^d \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} \\ &= 2 \sum_{k=0}^{\lfloor \frac{d}{2} \rfloor} \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\}, \end{aligned}$$

and $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$ is a multiple of 4.

Now suppose d and n are both even. Then $\lfloor \frac{d}{2} \rfloor$ is even since if $I \subseteq \mathbf{n}$ such that $|I|$ is even and $\sum_{i \in I} \alpha_i = \frac{d}{2}$, then $|\mathbf{n} \setminus I|$ is even and $\sum_{i \in \mathbf{n} \setminus I} \alpha_i = \frac{d}{2}$. Likewise, $|\mathcal{O}_{\frac{d}{2}}|$ is even. Since

$$\begin{aligned} \left\| \prod_{i=1}^n (1 - x^{\alpha_i}) \right\|_1 &= 2^n - 2 \sum_{k=0}^d \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} \\ &= 2^n - 2 \left(\sum_{k=0}^{\frac{d}{2}-1} \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} + \min\{|\mathcal{E}_{\frac{d}{2}}|, |\mathcal{O}_{\frac{d}{2}}|\} + \sum_{k=\frac{d}{2}+1}^d \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} \right) \\ &= 2^n - 4 \sum_{k=0}^{\frac{d}{2}-1} \min\{|\mathcal{E}_k|, |\mathcal{O}_k|\} + 2 \min\{|\mathcal{E}_{\frac{d}{2}}|, |\mathcal{O}_{\frac{d}{2}}|\}, \end{aligned}$$

this tells us that $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$ is a multiple of 4. □

The following result is not difficult or original, but we want to mention it in order to point out an oversight of Borwein and Ingalls. Bachman and Narici [BN66, Example 19.5, pp. 312-313] say without proof that the set of all polynomials with complex coefficients forms a normed algebra when the 1-norm is taken as the norm of the algebra. Implicit in this statement is the following result.

Theorem 2.5. *For all natural numbers m, n and $\alpha_1, \dots, \alpha_m, \beta_1, \dots, \beta_n$,*

$$\left\| \prod_{i=1}^m (1 - x^{\alpha_i}) \cdot \prod_{i=1}^n (1 - x^{\beta_i}) \right\|_1 \leq \left\| \prod_{i=1}^m (1 - x^{\alpha_i}) \right\|_1 \cdot \left\| \prod_{i=1}^n (1 - x^{\beta_i}) \right\|_1.$$

Proof. Let M, N and $a_0, \dots, a_M, b_0, \dots, b_N, c_0, \dots, c_{M+N}$ so that $\sum_{k=0}^M a_k x^k = \prod_{i=1}^m (1 - x^{\alpha_i})$, $\sum_{k=0}^N b_k x^k = \prod_{i=1}^n (1 - x^{\beta_i})$, and $\sum_{k=0}^{M+N} c_k x^k = \prod_{i=1}^m (1 - x^{\alpha_i}) \cdot \prod_{i=1}^n (1 - x^{\beta_i})$. So each $c_k = \sum_{i+j=k} a_i b_j$ and

$$\begin{aligned} \sum_{k=0}^{M+N} |c_k| &= \sum_{k=0}^{M+N} \left| \sum_{i+j=k} a_i b_j \right| \leq \sum_{k=0}^{M+N} \sum_{i+j=k} |a_i b_j| \\ &= \sum_{\substack{0 \leq i \leq M \\ 0 \leq j \leq N}} |a_i| \cdot |b_j| = \left(\sum_{i=0}^M |a_i| \right) \left(\sum_{j=0}^N |b_j| \right) \\ &= \left\| \prod_{i=1}^m (1 - x^{\alpha_i}) \right\|_1 \cdot \left\| \prod_{i=1}^n (1 - x^{\beta_i}) \right\|_1. \quad \square \end{aligned}$$

Theorem 2.5 tells us that $A_1(m + n) \leq A_1(m) \cdot A_1(n)$ for all $m, n \in \mathbf{N}$. In fact, if $m + n > 2$, then $A_1(m + n) < A_1(m) \cdot A_1(n)$, but we will not prove this since we have not been able to prove that the difference between $A_1(m + n)$ and $A_1(m) \cdot A_1(n)$ is big enough to be interesting.

In [BI94], Borwein and Ingalls give upper bounds on some values of $A_1(n)$. We are now able to see that two of their bounds are weaker than they could be. They say that $A_1(22) \leq 140$, $A_1(58) \leq 6268$, and $A_1(80) \leq 1,629,900$. But now we see that $A_1(80) = A_1(22 + 58) \leq A_1(22) \cdot A_1(58) \leq 140 \cdot 6268 = 877,520$. In fact, we do even better than this by explicitly computing the norm of the product of Borwein and Ingalls’s 22-factor and 58-factor pure products — it is 58,488. Borwein and Ingalls also say that $A_1(41) \leq 1348$, $A_1(59) \leq 7572$, and $A_1(100) \leq 41,947,220$. So $A_1(100) = A_1(41 + 59) \leq A_1(41) \cdot A_1(59) \leq 1348 \cdot 7572 = 10,207,056$. Explicitly computing the norm of the product of their 41-factor and 59-factor pure products, we find that this 100-factor pure product has a norm of 385,620. Hence,

	Borwein & Ingalls	new
$A_1(80) \leq$	1,629,900	58,488
$A_1(100) \leq$	41,947,220	385,620

3. A SIMPLE ALGORITHM

If one wants to determine whether there is an n -factor pure product having norm at most k , the obvious unsophisticated method to use is simply to keep substituting values for $\alpha_1, \dots, \alpha_n$ and see what norms result. Before our research, it was known that $A_1(n) \geq 2n$ for all $n \in \mathbf{N}$ and $A_1(n) = 2n$ for $n \leq 6$ and $n = 8$. So the obvious question to ask was: *Is $A_1(7) = 14$?* Borwein and Ingalls [BI94] said they computed extensively without finding any 7-factor pure product having norm 14, and they conjectured that $A_1(7) = 16$. As an example of a 7-factor pure product having norm 16, they presented $\prod_{i=1}^7 (1 - x^{\alpha_i})$ where $[\alpha_1, \dots, \alpha_7] = [1, 2, 3, 4, 5, 7, 11]$. Independently of each other and of Borwein and Ingalls, S. Maltby [M94a] and Walley [W94] searched all possible values of $\alpha_1, \dots, \alpha_7$ up to about 40 and found the following 7-lists for $[\alpha_1, \dots, \alpha_7]$ which yield a 7-factor pure product of norm 16. Previously, we had found all but two of these without computer assistance as

reported in Section 5.

- [1, 1, 2, 3, 4, 5, 7]
- [1, 2, 3, 4, 5, 7, 11]
- [1, 2, 3, 5, 7, 8, 13]
- [1, 3, 4, 5, 7, 8, 11]
- [1, 3, 4, 5, 7, 11, 17]
- [1, 3, 4, 7, 10, 11, 13]
- [1, 5, 6, 7, 8, 11, 13]
- [2, 3, 5, 7, 8, 11, 13]

Before our research, it was unknown whether any finite amount of searching could be considered exhaustive. The proof of Theorem 3.4 uses Bombieri and Vaaler’s [BV83] improvement of Siegel’s Lemma [S29]; [M69, pp. 32–33] to show that to find all possible norms of n -factor pure products, it suffices to check the norms for all values of $\alpha_1, \dots, \alpha_n$ up to $\sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$ (which is slightly less than $n^{\frac{3}{4}(n-1)}$). One could just use Siegel’s Lemma, but it yields a weaker bound of n^{n-1} .

Lemma 3.1. *Let $a_{1,1}, \dots, a_{n-1,n} \in \{1, 0, -1\}$. Then there exist integers x_1, \dots, x_n each of absolute value at most $\sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$, satisfying*

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ \vdots & \vdots & & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Proof. Let A denote the $(n-1) \times n$ matrix in the statement of the lemma. Theorem 1 in Bombieri and Vaaler’s paper [BV83] tells us that there exist integers x_1, \dots, x_n each of absolute value at most $\sqrt{\det(AA^T)}$ which solve this system. In the $(n-1) \times (n-1)$ matrix AA^T , each entry is a dot product $(a_{i,1}, \dots, a_{i,n}) \cdot (a_{j,1}, \dots, a_{j,n})$ of absolute value at most n . So Hadamard’s inequality tells us that $|\det(AA^T)| \leq n^{n-1}(n-1)^{\frac{n-1}{2}}$, and Bombieri and Vaaler’s result gives us the conclusion of the lemma. \square

Lemma 3.2. *Suppose A is an n -column matrix of rank $n-1$ and each entry of A is 1, 0, or -1 . Suppose there exist positive reals y_1, \dots, y_n so that $A\mathbf{y} = \mathbf{0}$. Then there exist positive integers x_1, \dots, x_n each at most $\sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$ satisfying $A\mathbf{x} = \mathbf{0}$.*

Proof. Since A has rank $n-1$, we know that

$$\{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}\} = \{t(y_1, \dots, y_n) : t \in \mathbf{R}\}.$$

Since y_1, \dots, y_n are all positive, this tells us that any x_1, \dots, x_n satisfying $A\mathbf{x} = \mathbf{0}$ are all positive, all negative, or all zero. Lemma 3.1 tells us that there are integers x_1, \dots, x_n not all zero and each of absolute value at most $\sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$ satisfying $A\mathbf{x} = \mathbf{0}$. If these x_1, \dots, x_n are all positive, then we have the conclusion of the lemma. Otherwise, these x_1, \dots, x_n are all negative, and we get the conclusion of the lemma by using $-x_1, \dots, -x_n$. \square

The following lemma may seem familiar to readers who have searched for positive systems of roots determining a root system. Other interesting connections of root systems to pure products of small norm are discussed in [M96] and [M97].

Lemma 3.3. *Suppose A is an n -column matrix of rank $n - r$ where $r \geq 2$ and each entry of A is 1, 0, or -1 . Suppose there exist positive reals y_1, \dots, y_n so that $A\mathbf{y} = \mathbf{0}$. Then there exists an n -column matrix A' of rank $n - r + 1$ such that each entry of A' is 1, 0, or -1 , and so that $\{(x_1, \dots, x_n) : A'\mathbf{x} = \mathbf{0}\} \subset \{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}\}$, and so that there exist positive reals y'_1, \dots, y'_n satisfying $A'\mathbf{y}' = \mathbf{0}$.*

Proof. Since A has rank $n - r$, we know that there exist $z_{1,1}, \dots, z_{r,n} \in \mathbf{R}$ so that

$$\{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}\} = \left\{ \sum_{i=1}^r t_i(z_{i,1}, \dots, z_{i,n}) : t_1, \dots, t_r \in \mathbf{R} \right\}.$$

That is, all solutions of the equation $A\mathbf{x} = \mathbf{0}$ are described by the formula

$$t_1(z_{1,1}, \dots, z_{1,n}) + t_2(z_{2,1}, \dots, z_{2,n}) + \dots + t_r(z_{r,1}, \dots, z_{r,n}) = (x_1, \dots, x_n).$$

We assume without loss of generality that $(z_{1,1}, \dots, z_{1,n}) = (y_1, \dots, y_n)$. So we have $z_{1,1}, \dots, z_{1,n} > 0$ and $(z_{1,1}, z_{2,1}), \dots, (z_{1,n}, z_{2,n})$ all lie in the half-plane in \mathbf{R}^2 where points have positive first coordinates. Since $(z_{2,1}, \dots, z_{2,n})$ is not a scalar multiple of $(z_{1,1}, \dots, z_{1,n})$, we know that the vectors $(z_{1,1}, z_{2,1}), \dots, (z_{1,n}, z_{2,n})$ are not all coincident. Let $k, l \in \mathbf{n}$ so as to maximise the angle between $(z_{1,k}, z_{2,k})$ and $(z_{1,l}, z_{2,l})$. Then for $i = 1, \dots, n$, let $z'_{1,i}, z'_{2,i}$ so that $z'_{1,i}(z_{1,k}, z_{2,k}) + z'_{2,i}(z_{1,l}, z_{2,l}) = (z_{1,i}, z_{2,i})$. Since every vector $(z_{1,i}, z_{2,i})$ lies in the sector bounded by $(z_{1,k}, z_{2,k})$ and $(z_{1,l}, z_{2,l})$, we know that $z'_{1,i}, z'_{2,i} \geq 0$ for all $i = 1, \dots, n$. Also, notice that $z'_{1,k} = z'_{2,l} = 1$ and $z'_{2,k} = z'_{1,l} = 0$.

Let $t_1, t_2 \in \mathbf{R}$. Let $t'_1 = t_1 z_{1,k} + t_2 z_{2,k}$ and $t'_2 = t_1 z_{1,l} + t_2 z_{2,l}$. Then for each $i = 1, \dots, n$,

$$\begin{aligned} t'_1 z'_{1,i} + t'_2 z'_{2,i} &= (t_1 z_{1,k} + t_2 z_{2,k}) z'_{1,i} + (t_1 z_{1,l} + t_2 z_{2,l}) z'_{2,i} \\ &= t_1 (z_{1,k} z'_{1,i} + z_{1,l} z'_{2,i}) + t_2 (z_{2,k} z'_{1,i} + z_{2,l} z'_{2,i}) \\ &= t_1 z_{1,i} + t_2 z_{2,i}. \end{aligned}$$

So

$$t'_1(z'_{1,1}, \dots, z'_{1,n}) + t'_2(z'_{2,1}, \dots, z'_{2,n}) = t_1(z_{1,1}, \dots, z_{1,n}) + t_2(z_{2,1}, \dots, z_{2,n}).$$

So we see that

$$\begin{aligned} &\{t_1(z_{1,1}, \dots, z_{1,n}) + t_2(z_{2,1}, \dots, z_{2,n}) : t_1, t_2 \in \mathbf{R}\} \\ &= \{t'_1(z'_{1,1}, \dots, z'_{1,n}) + t'_2(z'_{2,1}, \dots, z'_{2,n}) : t'_1, t'_2 \in \mathbf{R}\}. \end{aligned}$$

Let A' be the matrix A with one row appended, that row being $[a_1 \ \dots \ a_n]$ where $a_i = 0$ for $i = 1, \dots, n$, except for $a_k = 1$ and $a_l = -1$. Then

$$\begin{aligned} &\{(x_1, \dots, x_n) : A'\mathbf{x} = \mathbf{0}\} \\ &= \{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}, (a_1, \dots, a_n) \cdot (x_1, \dots, x_n) = 0\} \\ &= \{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}, x_k = x_l\} \\ &\subseteq \{(x_1, \dots, x_n) : A\mathbf{x} = \mathbf{0}\}. \end{aligned}$$

To prove the conclusion of the theorem, we just have to show that there are positive reals y'_1, \dots, y'_n such that $A'\mathbf{y}' = \mathbf{0}$. We have y_1, \dots, y_n so that $A\mathbf{y} = \mathbf{0}$ and

$$1(z_{1,1}, \dots, z_{1,n}) + 0(z_{2,1}, \dots, z_{2,n}) + \dots + 0(z_{r,1}, \dots, z_{r,n}) = (y_1, \dots, y_n).$$

So there exist t'_1 and t'_2 (in fact, $t'_1 = z_{1,k}$ and $t'_2 = z_{1,l}$) so that

$$t'_1(z'_{1,1}, \dots, z'_{1,n}) + t'_2(z'_{2,1}, \dots, z'_{2,n}) + 0(z_{3,1}, \dots, z_{3,n}) + \dots + 0(z_{r,1}, \dots, z_{r,n}) = (y_1, \dots, y_n).$$

Let $t' = \max\{t'_1, t'_2\}$ and let y'_1, \dots, y'_n so that

$$t'(z'_{1,1}, \dots, z'_{1,n}) + t'(z'_{2,1}, \dots, z'_{2,n}) + 0(z_{3,1}, \dots, z_{3,n}) + \dots + 0(z_{r,1}, \dots, z_{r,n}) = (y'_1, \dots, y'_n).$$

Since $t' \geq t'_1, t'_2$ and $z'_{1,1}, \dots, z'_{1,n}, z'_{2,1}, \dots, z'_{2,n} \geq 0$, we know that $y'_i \geq y_i$ for $i = 1, \dots, n$. Since y_1, \dots, y_n are all positive, this tells us that y'_1, \dots, y'_n are all positive. We also have $y'_k = t'z'_{1,k} + t'z'_{2,k}$ and $y'_l = t'z'_{1,l} + t'z'_{2,l}$. Since $z'_{1,k} = z'_{1,l} = 1$ and $z'_{2,k} = z'_{2,l} = 0$, this tells us that $y'_k = t' = y'_l$. So

$$(y'_1, \dots, y'_n) \in \{(x_1, \dots, x_n) : \mathbf{Ax} = \mathbf{0}, x_k = x_l\} = \{(x_1, \dots, x_n) : \mathbf{A}'\mathbf{x} = \mathbf{0}\}.$$

That is, $\mathbf{A}'\mathbf{y}' = \mathbf{0}$. □

Theorem 3.4. *If there is an n -factor pure product $\prod_{i=1}^n (1 - x^{\alpha_i})$ having norm k , then there is a pure product $\prod_{i=1}^n (1 - x^{\beta_i})$ of norm k or less having every $\beta_i \leq \sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$.*

Proof. Let $m = \sum_{i=1}^n \alpha_i$ and define $\mathcal{E}_0, \dots, \mathcal{E}_m$ and $\mathcal{O}_0, \dots, \mathcal{O}_m$ as in Lemma 2.1. For all $k \in \{0, \dots, m\}$, and every $I \in \mathcal{E}_k$ and $J \in \mathcal{O}_k$, define $A(I, J)$ to be the n -entry row matrix $[a_1 \ \dots \ a_n]$ where $a_i = 1$ for each $i \in I$, $a_j = -1$ for each $j \in J$, and the other entries are all zeroes. Let A be a matrix whose rows are all the $A(I, J)$'s. It is clear that $A\vec{\alpha} = \mathbf{0}$, and from Lemma 2.3 it is clear that $\|\prod_{i=1}^n (1 - x^{\beta_i})\|_1 \leq \|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$ for any β_1, \dots, β_n such that $A\vec{\beta} = \mathbf{0}$ since $\sum_{i \in I} \beta_i = \sum_{j \in J} \beta_j$ for all $I, J \subseteq \mathbf{n}$ such that $|I|$ is even, $|J|$ is odd, and $\sum_{i \in I} \alpha_i = \sum_{j \in J} \alpha_j$. It is easy to see that the rank of A is at most $n - 1$ since for any $c \in \mathbf{N}$, if $(\beta_1, \dots, \beta_n) = c(\alpha_1, \dots, \alpha_n)$, then each \mathcal{E}_k (respectively, \mathcal{O}_k) for $\prod_{i=1}^n (1 - x^{\beta_i})$ equals $\mathcal{E}_{\frac{k}{c}}$ (respectively, $\mathcal{O}_{\frac{k}{c}}$) for $\prod_{i=1}^n (1 - x^{\alpha_i})$, and, hence $A\vec{\beta} = \mathbf{0}$. Suppose the rank of A is $n - r$. We have just seen that r must be at least 1. Since $\alpha_1, \dots, \alpha_n$ are all positive and $A\vec{\alpha} = \mathbf{0}$, we can apply Lemma 3.3 $r - 1$ times to get a matrix B of rank $n - 1$ and positive reals y_1, \dots, y_n so that each entry of B is 1, 0, or -1 , $B\mathbf{y} = \mathbf{0}$, and so that $\{(x_1, \dots, x_n) : B\mathbf{x} = \mathbf{0}\} \subseteq \{(x_1, \dots, x_n) : \mathbf{Ax} = \mathbf{0}\}$. Let B' be the matrix B with redundant rows removed. That is, let B' be a matrix whose rows are $n - 1$ linearly independent rows of B . Now Lemma 3.2 gives us positive integers β_1, \dots, β_n (or x_1, \dots, x_n as they are called in the lemma) so that $B'\vec{\beta} = \mathbf{0}$, and each $\beta_i \leq \sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$. Of course, $B'\vec{\beta} = \mathbf{0}$ implies $B\vec{\beta} = \mathbf{0}$ which implies $A\vec{\beta} = \mathbf{0}$, and this tells us that $\|\prod_{i=1}^n (1 - x^{\beta_i})\|_1 \leq \|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$. □

Obviously, the proof of Theorem 3.4 would be simpler if we could assume that for any n -factor pure product of minimal norm, the corresponding matrix A in Theorem 3.4 had rank $n - 1$. In that case, we would only need Lemma 3.2. However, the matrices in question can have rank $n - 2$ when $n \in \{2, 3, 4, 6\}$. It is somewhat surprising that this happens for a case as large as $n = 6$, and it would be very surprising if it happened for larger n , but we have no proof that it is impossible.

Theorem 3.4 shows that one way to determine $A_1(n)$ is to calculate the norm of every pure product $\prod_{i=1}^n (1 - x^{\alpha_i})$ where every $\alpha_i \leq \sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}}$. The

number of these pure products is

$$\binom{\left\lfloor \sqrt{n^{n-1}(n-1)^{\frac{n-1}{2}}} \right\rfloor + n - 1}{n} \approx \binom{\lfloor n^{\frac{3}{4}n} \rfloor}{n}.$$

Clearly, this is an impractical number of cases to check for any n for which $A_1(n)$ is not already known.

4. A BETTER SIMPLE ALGORITHM

Another simple-minded algorithm which eventually runs out of cases to check is a straightforward application of Lemma 2.3. For any $n \in \mathbf{N}$, there are only finitely many ways to choose \mathcal{P} in Lemma 2.3. Theoretically, one could check all possible choices of \mathcal{P} , solve for the corresponding pure products, and check their norms. Essentially, this is the same as checking all possible choices of the matrix A in Theorem 3.4, solving the associated systems $A\vec{\alpha} = \mathbf{0}$, and determining the associated norms $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$. The least such norm will be $A_1(n)$. Since these systems always have rank at most $n - 1$, it suffices to check all possible choices of up to $n - 1$ elements of \mathcal{P} , or, equivalently, all possible choices of up to $n - 1$ linearly independent rows of A . In [M96], the author shows that this means that this algorithm has to check at most

$$\binom{\frac{3^n - (-1)^n}{4}}{n - 1} < \binom{3^{n-1}}{n - 1}$$

cases, which is a big improvement on the algorithm in Section 3, even after we take into consideration the fact that each of these cases involves solving a system of linear equations.

5. A MORE SOPHISTICATED ALGORITHM

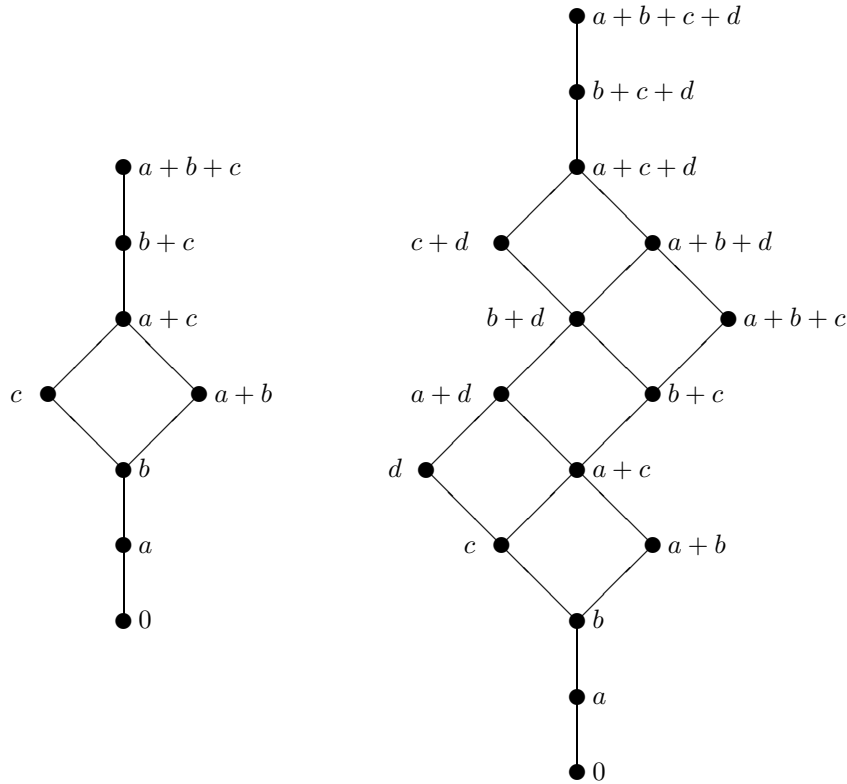
In this section, we describe a more substantial algorithm which finds all pure products of any given number of factors n and of norm up to any given value k . Our computations using this algorithm have determined that $A_1(n) > 2n$ for $n = 7, 9, 10, 11$, and that the list of 7-lists given at the start of Section 3 yielding a pure product of norm 16 is complete (up to multiplication of all the entries in the list by a constant).

So far we have written all our pure products as $\prod_{i=1}^n (1 - x^{\alpha_i})$. Much of what follows is easier to read if we rename the exponents so as to eliminate subscripts. So we use some symbols interchangeably as follows: $a = \alpha_1$, $b = \alpha_2$, *et cetera*. Also, we will always label the exponents so that $\alpha_1 \leq \alpha_2 \leq \alpha_3 \leq \dots$. That is, $a \leq b \leq c \leq \dots$.

For $I \subseteq \mathbf{n}$, we will call $\sum_{i \in I} \alpha_i$ a $|I|$ -sum. Also, we say that $\sum_{i \in I} \alpha_i$ is an *even-sum* if $|I|$ is even, and an *odd-sum* if $|I|$ is odd. We want to see what simultaneous equalities between even-sums and odd-sums are possible so that we can construct the corresponding set \mathcal{P} as described in Lemma 2.3.

Using only the information that $0 < a \leq b \leq c \leq \dots$, we can say something about how sums of subsets of $\{a, b, c, \dots\}$ compare with each other. For instance, it is clear that $a < a + b < a + b + c$ and so on. Another example is that $a + c \leq b + d$, so $a + c < a + b + d$. In fact, for each $n \in \mathbf{N}$, we can impose a partial ordering on the subsets of \mathbf{n} so that for any $I, J \subseteq \mathbf{n}$, we say that $I < J$ if we know that $\sum_{i \in I} \alpha_i \leq \sum_{j \in J} \alpha_j$ for all valid values of $\alpha_1, \dots, \alpha_n$ (that is, for all $\alpha_1, \dots, \alpha_n \in \mathbf{N}$

such that $0 < \alpha_1 \leq \alpha_2 \leq \dots \leq \alpha_n$). We call this ordering the *basic partial ordering*. For instance, the following diagrams show the basic partial orders for the cases $n = 3$ and $n = 4$.



In the case where $n = 3$, we see that for any valid values of a , b , and c , one of the following two statements must be true:

$$0 < a \leq b < a + b \leq c < a + c \leq b + c < a + b + c,$$

$$0 < a \leq b \leq c < a + b \leq a + c \leq b + c \leq a + b + c.$$

This makes it clear that there are not many ways that an even-sum can equal an odd-sum. In fact, the only such equality that is possible is $a + b = c$. What our program does is to determine all such equalities while determining all possible ways that the sums could be ordered. Essentially, this is the same as determining all possible linear extensions of the partial orders described above except that, of course, the extensions we are interested in are not strictly linear. That is, in a linear extension, of any two elements, one is greater; but we need to consider what equalities can occur also. So, for instance, the orderings we consider in the case $n = 3$ are not just the two linear extensions

$$0 < a < b < a + b < c < a + c < b + c < a + b + c,$$

$$0 < a < b < c < a + b < a + c < b + c < a + b + c.$$

Each of these orderings includes seven “<” symbols. The program must consider what happens each time a “<” is replaced by an “=”. We will call the resulting sequences *almost-linear extensions*. For instance, when $n = 2$, the basic partial

order is given by $0 < a < b < a + b$ which is its own only linear extension. So all the almost-linear extensions of the basic partial order in the case $n = 2$ are

$$0 < a < b < a + b,$$

$$0 < a < b = a + b,$$

$$0 < a = b < a + b,$$

$$0 < a = b = a + b,$$

$$0 = a < b < a + b,$$

$$0 = a < b = a + b,$$

$$0 = a = b < a + b,$$

$$0 = a = b = a + b.$$

For the present, we will ignore the fact that all but two of these almost-linear extensions include comparisons which are invalid. (We cannot have $0 = a$ or $b = a + b$ since $a > 0$.)

One can determine all possible norms of n -factor pure products by associating them with almost-linear extensions. For instance, if a, b, c satisfy $0 < a = b < a + b < c < a + c = b + c < a + b + c$, then we know that $\|(1 - x^a)(1 - x^b)(1 - x^c)\|_1 = 8$ because, with the exponents in strictly increasing order, $(1 - x^a)(1 - x^b)(1 - x^c) = 1 - 2x^a + x^{a+b} - x^c + 2x^{a+c} - x^{a+b+c}$. Another example is that if a, b, c satisfy $0 < a < b < a + b = c < a + c < b + c < a + b + c$, then we know that $\|(1 - x^a)(1 - x^b)(1 - x^c)\|_1 = 6$ because, with the exponents in strictly increasing order, $(1 - x^a)(1 - x^b)(1 - x^c) = 1 - x^a - x^b + x^{a+c} + x^{b+c} - x^{a+b+c}$. By generating all possible almost-linear extensions of a basic partial order for a particular n , we can determine all possible norms of n -factor pure products. Of course, one only wants to consider almost-linear extensions that make sense. For instance, one wants to reject any almost-linear extension that includes both the comparisons $a = b$ and $a + c < b + c$. We will address this issue presently.

Our program begins its execution by asking the user for a number of factors n , and a target norm k . Then the program generates almost-linear extensions of the basic partial order for the given value of n one term at a time, backtracking when necessary. That is, in each iteration, the program appends one term to those it has chosen so far in the almost-linear extension. After the initial term 0, each term consists of a sum preceded by either “<” or “=”. For instance, every extension begins with “ $0 < a$ ”, and there are two terms that can follow this: “ $= b$ ” and “ $< b$ ”. When the program finds that its choices so far cannot be an initial segment of an almost-linear extension of norm k or less, it backtracks by rejecting its last choice and replacing it with a different choice. If no other valid choices remain, the program tries to replace its choice one position earlier in the extension, and so on.

For values of n for which $A_1(n)$ is not already known, the time required to generate all almost-linear extensions of the basic partial order is prohibitive. As the program constructs an almost-linear extension, it rejects choices that cannot lead to an almost-linear extension of norm k or less, choices that are inconsistent with previous choices (such as having $a = b$ and $a + c < b + c$ in the same almost-linear extension), and redundant choices. The program also recognises when the choices it has already made completely determine the remainder of the almost-linear extension. We now describe the algorithm the program follows for these purposes.

In constructing the almost-linear extensions, the program uses four main data structures holding the following information: the almost-linear extension currently under consideration (to the extent that it has been determined so far); for each term, a list of all the terms that are eligible to follow it; for each sum in the basic partial order, a count of how many sums in the basic partial order must be added to the extension before this one is eligible; and a matrix R so that $R\vec{\alpha} = \mathbf{0}$. We now state the steps in the algorithm. After stating the whole algorithm, we will elaborate on some parts of it.

Step 1. Append another term to the end of the almost-linear extension.

Step 2. If the term in Step 1 included an "=", update R .

Step 3. If R has rank $n - 1$, then the extension is completely determined so go to Step 6.

Step 4. Construct the list of terms that could be next in the extension. Reject from the list terms of the following types:

- (i) any "<"-term if "<" would make the norm too big.
- (ii) any term whose sum is greater than another sum in the list.

If we find in this list a sum which is less than the last sum in the extension, go to Step 7.

Step 5. If we have determined half the extension, then we have determined the entire extension; otherwise, go to Step 1.

Step 6. To get to this step, the extension must be completely determined. Calculate the resulting norm. If it is small enough to be interesting, report the norm and how it is achieved (the extension and R).

Step 7. Remove the last term from the extension. If we have something else to replace it with, go to Step 1. Otherwise, repeat this step. The program terminates when this step backtracks all the way to the start of the extension.

This is not a complete description of our computer program. We used several tricks in writing the program which are vital for making the program as fast as it is, but which are not of much theoretical interest. For instance, in Step 4, when we compare all sums in the list, we remember which ones are equal so that in the next iteration of Step 1, we append not just one term, but also all the sums from the list which are equal to the sum in that term. We also will not describe the simple rule we used to make sure that equalities are used in only one permutation. For instance, the following two initial segments of an almost-linear extension are superficially different, and our program has distinct representations for them, but we do not want them to lead to distinct branches in the search tree since they are algebraically equivalent.

$$0 < a < b < a + b = c,$$

$$0 < a < b < c = a + b.$$

We now elaborate on some of these steps in order of complexity.

Step 5. This is essentially explained in the proof of Lemma 2.2.

Step 2. Suppose the term appended in Step 1 was " $= \sum_{i \in I} \alpha_i$ " and the sum in the previous term was $\sum_{j \in J} \alpha_j$. Then let a_1, \dots, a_n equal zero except that $a_i = 1$

for each $i \in I \setminus J$ and $a_j = -1$ for each $j \in J \setminus I$. Then $(a_1, \dots, a_n) \cdot (\alpha_1, \dots, \alpha_n) = \sum_{i \in I} \alpha_i - \sum_{j \in J} \alpha_j = 0$. We append the row $[a_1 \ \dots \ a_n]$ to the matrix R and then put R in reduced row-echelon form. So R remains a matrix so that $R\vec{\alpha} = \mathbf{0}$.

Step 4. The list of terms that could be next in the extension is a list of terms whose sums have not yet appeared in the extension, but all of whose predecessors in the basic partial order *have* already appeared in the extension. Actually, it may be a bit of an exaggeration to say the program “constructs” this list every time it executes Step 4. The program keeps track of appropriate data about the basic partial order and how many immediate predecessors of each sum have already been used so that updating this list in each iteration takes very little time. For each sum that could be next in the extension, we put two terms in the extension — one with “<” and one with “=” — unless a “<” at this point would force the norm of the extension to be too big, in which case we just use the “=”-term. This requires a relatively lengthy explanation, so we will elaborate on the rest of Step 4 first.

We compare all the sums in the list to each other as well as to the last sum in the extension using the matrix R . For instance, the equalities that have appeared so far in the extension, and hence are represented in R , may tell us that $b + c + f = 3b + 5a$ and $d + e = 2b + 5a$. So if both these sums appeared in the list of potential next terms, we would throw out of the list the term(s) in which $b + c + f$ appears, since any term with $d + e$ would have to appear first in the extension. If we find that there is a sum in the list which is equal to the last sum in the extension, then we throw out of the list all terms with sums not equal to the last sum in the extension. If we find that there is a term in the list whose sum is less than the last term in the extension, then we know that this branch of the search tree cannot lead to a complete algebraically valid almost-linear extension and we jump to Step 7 to backtrack.

The matrix R is constructed according to equalities that appear in the extension, but one can also glean important information from inequalities in the extension. For instance, if we had “ $g < b + c + e$ ” in the extension, and both $a + g$ and $a + b + c + e$ appearing in the list of possible next terms, we would reject from the list the term(s) using $a + b + c + e$ since we would need a term with $a + g$ to appear first.

Another effective measure used by the program is to keep track of how big or small b can be in terms of a . Theoretically, the matrix R might reduce sums to expressions in any of the variables, but experimentally we have found that almost all significant branches of the search tree have R reducing sums to expressions in a and b . For instance, equalities represented in R might tell us that $c = a + b$ and $d = 3a + b$. Now suppose the extension includes the relations $d < b + c < a + d$. This tells us that $3a + b < a + 2b < 4a + b$ and, hence, $2a < b < 3a$. If $a + b + c$ appeared in the list of potential next terms, any extension that we could get from this point would include the algebraic relation $a + b + c \geq a + d$, which we know implies $2a + 2b \geq 4a + b$, which implies $2a \geq b$, contradicting our bounds on b in terms of a . So if this happened, we would know that we could not construct a complete algebraically valid extension from this point, and we would jump to Step 7 to backtrack.

The preceding two paragraphs describing the use of inequalities to prune the search tree were not part of our original plan for Step 4, but the empirical evidence of initial runs of the program made it clear that the absence of these rules was causing the program to waste a lot of time in futile searches for algebraically

valid completions of extensions which one could prove had no algebraically valid completions.

We now describe what we meant by a “<” making the norm too big. The rule is easy to apply, but the underlying mathematics is somewhat involved.

Theorem 5.1. *If $\alpha_1, \dots, \alpha_n \in \mathbf{N}$, then*

$$\sum_{\substack{I \subseteq \mathbf{n} \\ |I| \text{ odd}}} \left(\sum_{i \in I} \alpha_i \right)^k = \sum_{\substack{J \subseteq \mathbf{n} \\ |J| \text{ even}}} \left(\sum_{j \in J} \alpha_j \right)^k$$

for all $k < n$. □

Dorwart and Brown [DB37, p. 625, item 5] attribute Theorem 5.1 to Escott. This author [M96] has found a new proof of this theorem which shows that the identity holds for $\alpha_1, \dots, \alpha_n$ in any ring. This result gives us an easy way to construct multigrades of size 2^{n-1} and degree $n - 1$. Furthermore, for each occurrence of I, J in Theorem 5.1 so that $\sum_{i \in I} \alpha_i = \sum_{j \in J} \alpha_j$, we can reduce the size of the multigrade by 1 since eliminating these two sums obviously leaves a multigrade of the same degree. With Lemma 2.3, this observation yields the following corollary (cf. [BI94, Proposition 10]; [M96]).

Corollary 5.2. *If there is an n -factor pure product of norm k , then there is a multigrade of degree $n - 1$ and size $\frac{k}{2}$. □*

Steinig [S71, Theorem A] says that the following theorem of Laguerre [L98, p. 28], was first proven correctly by Pólya [P13]. It requires some effort to piece together a proof from the literature; the pieces are collected and presented coherently in [M96].

Corollary 5.3. *Let $a_1 \leq \dots \leq a_k$ and $b_1 \leq \dots \leq b_l$ be natural numbers with no $a_i = b_j$. Let (c_1, \dots, c_{k+l}) be the permutation of $(a_1, \dots, a_k, b_1, \dots, b_l)$ so that $c_1 \leq \dots \leq c_{k+l}$. For $i = 1, \dots, k + l$, let*

$$s_i = |\{j : j \leq i, c_j \in A\}| - |\{j : j \leq i, c_j \in B\}|$$

and let

$$I = \{i : 2 \leq i \leq k + l - 1, (s_{i-1}, s_i, s_{i+1}) \in \{(-1, 0, 1), (1, 0, -1)\}\}.$$

Then there are no more than $|I|$ positive real values of t for which

$$\sum_{i=1}^k a_i^t = \sum_{i=1}^l b_i^t. \quad \square$$

One implication of this corollary is that a multigrade of degree d must have size at least $d + 1$. (Steinig [S71] attributes this result to Bastien [B13].) That is, if we have $a_1, \dots, a_k, b_1, \dots, b_k$ so that no $a_i = b_j$ and

$$\sum_{i=1}^k a_i^t = \sum_{i=1}^k b_i^t$$

for $t = 1, \dots, d$, then $k \geq d + 1$. Moreover, if $a_1 < b_1$, $a_1 \leq \dots \leq a_k$, $b_1 \leq \dots \leq b_k$, and $k = d + 1$, then

$$a_1 < b_1 \leq b_2 < a_2 \leq a_3 < b_3 \leq b_4 < a_4 \leq a_5 < \dots$$

If one is only interested in precisely this rule, a simple proof is repeated in [BI94, Corollary 2, pp. 8–9], but even if this pattern is not followed exactly, Corollary 5.3 gives us a rule telling us how big k has to be in terms of how much the order of the a_i 's and b_i 's differs from this pattern. The way in which these numbers correspond to the pure product problem is that, for any pure product $\prod_{i=1}^n (1 - x^{\alpha_i})$, if one constructs \mathcal{P} as in Lemma 2.3, then the a_i 's and b_i 's are the even-sums and odd-sums, respectively, which do not appear in \mathcal{P} , so the corresponding norm is $2k$. And, for instance, if we want an almost-linear extension corresponding to an n -factor pure product of norm $2n$, it could begin with

$$0 < a < b < c = a + b,$$

but it could not begin with

$$0 < a < b < c < \dots$$

This is what we mean by a “ $<$ ” making the norm too big. This completes our description of the algorithm.

When we first developed this algorithm, we used it to work through the $n = 7$ case by hand to prove that $A_1(7) = 16$. This proof was 40 pages long. We started with the very long case $n = 7$ because it was the least n for which $A_1(n)$ was not already known. Afterward, S. Maltby [M94a] determined all possible ways of achieving a norm of $2n$ for $n = 1, \dots, 6$ following the same method. When we finished the computer program, it verified these results. In particular, it verified the 40-page proof in 5 seconds.

The following table shows all the results our program has produced. For $n = 3, \dots, 11$, the table gives a lower bound on $A_1(n)$, the least norm of an n -factor pure product found by the program, and the values of $[\alpha_1, \dots, \alpha_n]$ found by the program which realise that norm for $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1$. Except for the cases $n = 10$ and $n = 11$, our program has verified that the table's list of values of $[\alpha_1, \dots, \alpha_n]$ realising the given norm is complete. In this table, a and b are parameters for which one can substitute any natural numbers. Asterisks indicate results which have not appeared previously in the literature, although some have been discovered independently. In particular, S. Maltby [M94a] and Walley [W94] simply used computers to generate 7-lists and checked their norms to discover all the 7-lists in the table after we discovered all but two of them by hand. S. Maltby [M94a]; [M94b] also found some of the other lists in the table. The lists not marked with asterisks appear in Borwein and Ingalls's paper [BI94]. Borwein [B94] says they found some of the other lists in the table but only recorded the ones in their paper. We did not run the program long enough to get any results for 12-factor pure products, but S. Maltby [M94b] found that $\|\prod_{i=1}^{12} (1 - x^{\alpha_i})\|_1 = 36$ for $[\alpha_1, \dots, \alpha_{12}] = a[1, 2, 3, 5, 7, 8, 9, 11, 13, 17, 19, 31]$, which is different from Borwein

and Ingalls's example $a[1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 17]$.

n	$A_1(n) \geq$	$A_1(n) \leq$	$[\alpha_1, \dots, \alpha_n]$
3	6	6	$[a, b, a + b]^*$
4	8	8	$[a, b, a + b, a + 2b]^*$
5	10	10	$a[1, 2, 3, 5, 7]$ $a[1, 3, 4, 5, 7]^*$
6	12	12	$[a, b, a + b, a + 2b, a + 3b, 2a + 3b]^*$ $a[1, 3, 4, 5, 7, 11]$
7	16*	16	$a[1, 1, 2, 3, 4, 5, 7]^*$ $a[1, 2, 3, 4, 5, 7, 11]$ $a[1, 2, 3, 5, 7, 8, 13]^*$ $a[1, 3, 4, 5, 7, 8, 11]^*$ $a[1, 3, 4, 5, 7, 11, 17]^*$ $a[1, 3, 4, 7, 10, 11, 13]^*$ $a[1, 5, 6, 7, 8, 11, 13]^*$ $a[2, 3, 5, 7, 8, 11, 13]^*$
8	16	16	$a[1, 2, 3, 5, 7, 8, 11, 13]$ $a[2, 3, 5, 7, 8, 11, 13, 18]^*$ $a[2, 3, 5, 7, 8, 11, 13, 19]^*$
9	20*	20	$a[1, 2, 3, 4, 5, 7, 9, 11, 13]$ $a[1, 2, 3, 5, 7, 8, 9, 11, 13]^*$ $a[1, 2, 3, 5, 7, 8, 11, 13, 19]^*$ $a[1, 4, 5, 6, 7, 9, 11, 13, 17]^*$ $a[2, 3, 5, 7, 8, 11, 13, 17, 19]^*$
10	24*	24	$a[1, 2, 3, 4, 5, 7, 9, 11, 13, 17]$
11	24*	28	$a[1, 2, 3, 5, 7, 8, 9, 11, 13, 17, 19]$

How many steps are required by this algorithm to determine whether there is an n -factor pure product of norm k or less? It seems hopelessly complicated to answer this question accurately, so we will just get an upper bound and provide some empirical data. For n -factor pure products, the basic partial order has 2^n elements. A partial order on 2^n elements has at most $2^n!$ linear extensions and $2^n!2^{2^n-1}$ almost-linear extensions. This last number is an upper bound on the number of cases the program evaluates, but the following empirical data show that the program actually does much better than this. The following table lists the number of iterations required by the program to determine all n -lists $[\alpha_1, \dots, \alpha_n]$ so that $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1 = 2n$. One can see that, at least up to $n = 11$, the number

of iterations required to find all n -lists $[\alpha_1, \dots, \alpha_n]$ so that $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1 = 2n$ is very roughly $n!$.

n	iterations
3	12
4	39
5	171
6	790
7	4692
8	23,358
9	138,335
10	1,189,069
11	19,241,795

Determining all n -lists $[\alpha_1, \dots, \alpha_n]$ so that $\|\prod_{i=1}^n (1 - x^{\alpha_i})\|_1 = 2n + 2$ required 10,750 iterations for $n = 7$ and 508,820 iterations for $n = 9$. The program computed that $20 < A_1(10) \leq 24$. Because of Theorem 2.4, this tells us that $A_1(10) = 24$. We ran the program to find all 10-lists so that $\|\prod_{i=1}^{10} (1 - x^{\alpha_i})\|_1 = 24$, but after millions of iterations (*i.e.* a couple of days), the program seemed to have covered only a fraction of its search tree, so we terminated execution. So even though the algorithm provides a way to calculate all n -factor pure products of norm k for any $n \in \mathbf{N}$, the algorithm may not be practical when $k > 2n$.

ACKNOWLEDGEMENT

The author wishes to thank Samuel J. Maltby for several very useful suggestions.

REFERENCES

- [BN66] George Bachman and Lawrence Narici, *Functional Analysis*, Academic Press (1966). MR **36**:638
- [B13] L. Bastien, *Impossibilité de $u + v = \frac{3}{2}x + y + z$* , *Sphinx-Oedipe* 8 (1913), 171–172.
- [BV83] E. Bombieri and J. Vaaler, “On Siegel’s Lemma”, *Invent. Math.* **73** (1983), 11–32. MR **85g**:11049a
- [B94] Peter Borwein, personal communication.
- [BI94] Peter Borwein and Colin Ingalls, “The Prouhet-Tarry-Escott Problem Revisited”, *Enseign. Math. (2)* **40** (1994), 3–27. MR **95d**:11038
- [DB37] H.L. Dorwart and O.E. Brown, “The Tarry-Escott Problem”, *M.A.A. Monthly* **44** (1937), 613–626.
- [ES58] P. Erdős and G. Szekeres, “On the Product $\prod_{k=1}^n (1 - z^{\alpha_k})$ ”, *Acad. Serbe Sci. Publ. Inst. Math.* **13** (1959), 29–34. MR **23**:A3721
- [L98] E. Laguerre, *Oeuvres*, vol. 1, Gauthier-Villars, Paris (1898).
- [M94a] Samuel J. Maltby, “Some Optimal Results Related to the PTE Problem”, preprint.
- [M94b] Samuel J. Maltby, personal communication.
- [M69] L.J. Mordell, *Diophantine Equations*, Academic Press (1969). MR **40**:2600
- [M96] R. Maltby, *Pure Product Polynomials of Small Norm*, Ph.D. dissertation, Simon Fraser University (1996).
- [M97] R. Maltby “Root Systems and the Erdős-Szekeres Problem”, submitted to *Acta Arithmetica*.
- [P13] G. Pólya, “Sur un théorème de Laguerre”, *C.R. Acad. Sci. Paris* **156** (1913), 996–999.
- [S29] C.L. Siegel, “Über einige Anwendungen diophantischer Approximationen”, *Abhandlungen der Preussischen Akademie der Wissenschaften, Physikalisch-Mathematische Klasse, Nr. 1* (1929), (*Gesammelte Abhandlungen I*, 209–266, Springer-Verlag, 1966).

- [S71] John Steinig, "On some rules of Laguerre's, and systems of equal sums of like powers",
Rome U., Rend. Mat. (6) **4** (1971), 629–644. MR **46**:8972
- [W94] David Walley, personal communication.

CENTRE FOR EXPERIMENTAL AND CONSTRUCTIVE MATHEMATICS, SIMON FRASER UNIVERSITY,
BURNABY, BC, CANADA V5A 1S6
E-mail address: `maltby@cecm.sfu.ca`